COMPUTATIONAL METHODS AND SOFTWARE SYSTEMS FOR DYNAMICS AND CONTROL OF LARGE SPACE STRUCTURES

K. C. Park, C. A. Felippa, Charbel Farhat
J. D. Downer, J. C. Chiou and W. K. Belvin†
Center for Space Structures and Controls, Campus Box 429
College of Engineering and Applied Science
University of Colorado, Boulder, CO 80309

ABTRACT

The deployment, assembly and mission-oriented maneuvering of space structures in orbit will trigger large motions of flexible, truss-type structures. In addition, the presence of on-board controls both for attitude stabilization and specified vibration tolerance requirements may further complicate the dynamic behavior of the orbiting structures. Because of safety and cost considerations, the dynamic response of the combined structural and control systems must be predicted reliably. This need can only be met through the development of reliable and efficient simulation capabilities, since there is general agreement that on-orbit experiments should be limited because of cost, time and facility constraints.

The long-term objective of this research effort is to develop *a next-generation computer simulator* for the dynamics and control of large space structures. The simulator will be based on integrating four research thrusts: a new multibody dynamics formulation methodology, modeling capabilities in long/slender truss-beam components with realistic joints, efficient computational procedures that can be implemented either in sequential or concurrent computers, and prototype simulation modules that can be easily processed into a modern large-scale engineering software system such as the NASA/CSM testbed.

†On academic leave from Structures and Dynamics Division, NASA/Langley Research Center.

RESEARCH THRUST

Logn-Term Research Thrust:

- Develop a Next-Generation Computational Capability for Dynamics and Control of Large Space Structures.

Current Research Thrusts:

- Dynamics of Flexible Beams for Large Motions
- Computational Methods for Large Rotational Motions and Constraints
- Design of a *CONCURRENT* NICE (C-NICE) Architecture.
- Concurrent Computations via FORCE on CRAY-II and Alliant

CURRENT RESEARCH THRUSTS

In order to accomplish the research objectives set forth, we have carefully defined the scope of the present research effort as follows.

1. A multibody dynamics formulation that can represent combined large rigid and flexible motions, and that incorporates an objective way of modeling beams in hierarchical order from Euler-Bernoulli elements to bending-transverse shear and bending-torsion coupling.

2. A set of computational procedures that can treat large rotational motions, kinematic and dynamic constraints, contact-impact phenomena and computational stabilization for momentum and energy conservation, when necessary; these procedures must be able to perform well under both sequential and parallel computing environments.

3. A prototype *concurrent* NICE that preserves the program modularity between the processors and data management tasks and that allows the adoption of loosely coupled multiprocessors so as to achieve a smooth adaptation to parallel computing of processors developed under the sequential computing environment.

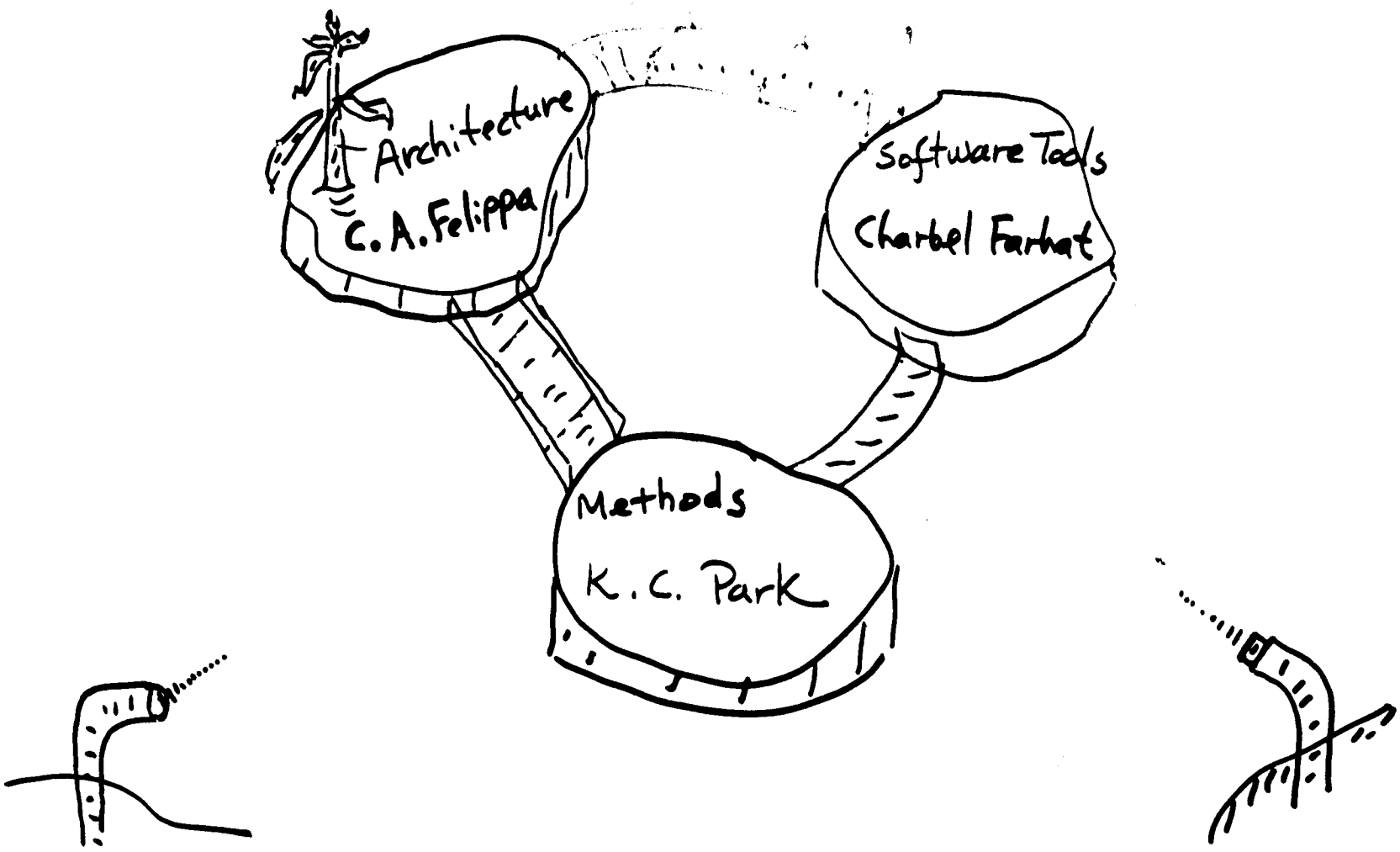
4. Implementation of the above three capabilities into a research-mode processor and validation of the module by the use of FORCE on Alliant and CRAY-II.

THIS PAGE LEFT BLANK INTENTIONALLY

OVERALL RESEARCH APPROACH

- Emerging Computer Hardware and Software Transportability Should Dictate Software Architacture Design and Implementation.
- Software Architecture Advances Should Influence New Formulations and New Methods Development.
- Needs for New Capability and its Implemantability Should Guide New Formulations *From the Outset.*
- Software Modularity Should Be a Key Aspect of Methods Development.

Overall Approach
(An Experiment in A University Setting)



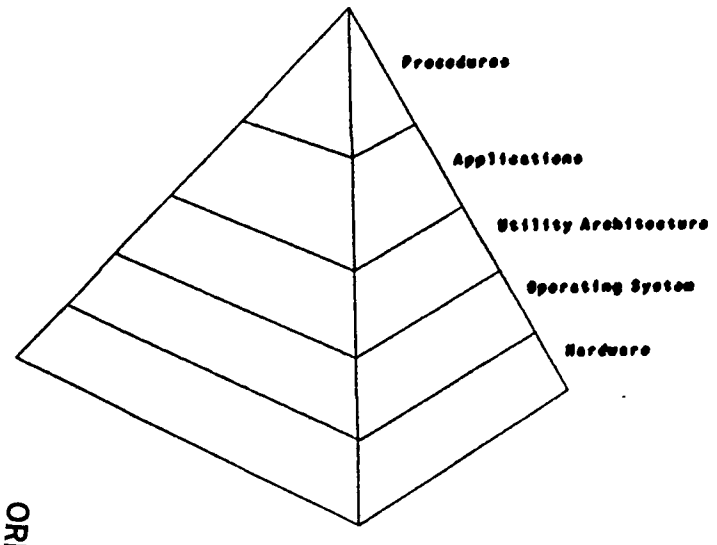
OVERALL RESEARCH APPROACH

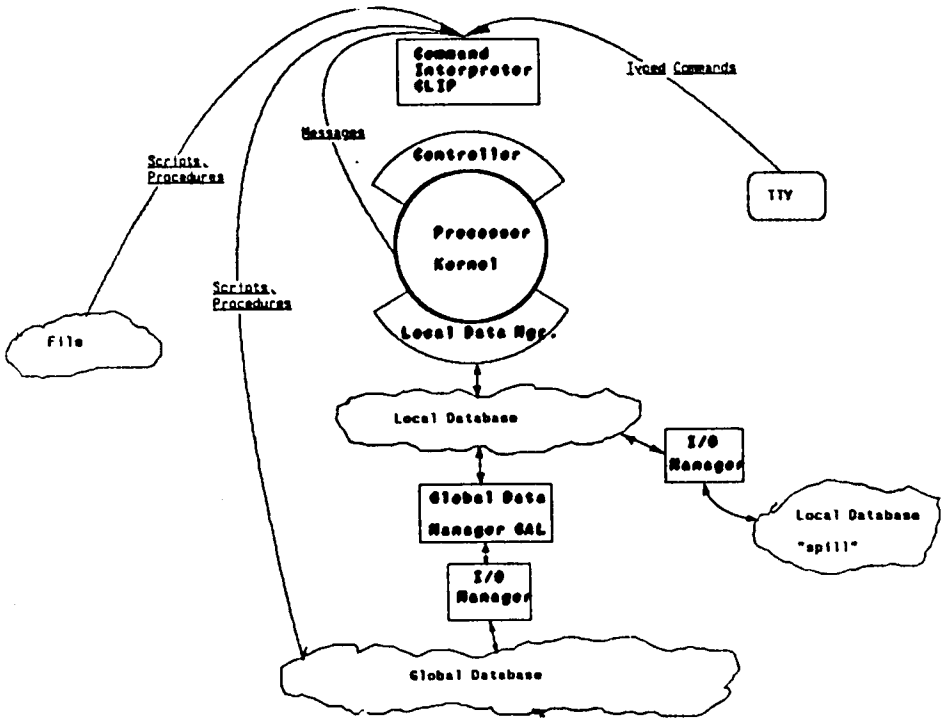
In traditional structural mechanics research, the needs for new/improved analysis capabilities are first identified. Next, the necessary formulations are carried out. The methods development is then undertaken to implement the formulations. Finally, the implemented programs are made available to the analyst for testing and applications.

In our approach to computational structural mechanics, we establish the needs for new/improved analysis capabilities. In our case, it is a general multibody dynamics and control simulation capability for parallel computers. First, we survey available analysis modules –i.e., direct time integration module. We then identify new software modules to be constructed. Only after decisions are made as to what new modules need to be developed, we will begin the necessary formulations and methods development.

Upon completion of new software modules, we then rely on *a command language* to integrate both the available and new software modules into a high-level analysis procedure. It is important that this procedure developments require both non-numerical and numerical algorithms. For example, many aspects of partitioned analysis procedures can be implemented during the procedure construction stage rather than in the analysis module development stage.

C-NICE: BACKGROUND





C-NICE: BACKGROUND

An applications-software architecture may be visualized as a layer of software that helps the development of applications by "cushioning" the interface of the programs with the operating system (see Figure).

A utility architecture is supplied in the form of tools or products that may be used selectively. Utilities architectures are most effective when the tools are in the public domain and subject to scrutiny.

NICE is an utility architecture developed to support computational mechanics applications and in particular finite element computations. This architecture allows the development of individual program modules called processors, and the database coupling of processors to form program networks.

Motivation. The original development of NICE was prompted by the idea of combining the power of logically related but separate programs. The envisioned applications were product design, loosely-coupled research, and the analysis of coupled problems. Additional motivation now arising from concurrent computation is the need to carry out time-critical simulations.

Components of the NICE architecture that enforce network operational compatibility are the command language interpreter CLIP and the global database manager GAL (see Figure). On sequential machines this form of architecture is stable, well developed and gaining acceptance.

Concurrent NICE. The question arises as to the architectural support of large-scale finite element computations on the newer concurrent processing machines. The development of C-NICE addresses that question.

C-NICE: OBJECTIVE

- *General objective*: to develop generic tools for the implementation of application program networks on the next generation of concurrent supercomputers.

- *Specific objective*: to support the development of structural dynamics and controls programs for simulation of large space structures on existing multiprocessors.

C-NICE: APPROACH

- Implement finite element programs on new concurrent machine architectures that span a broad spectrum.

- Keep in mind a wide range of needs, including structural dynamics, controls, and other coupled field problems.

- Develop processor mapping and assignment methods, new data structures, programming language requirements.

- Assess performance, programming difficulties, project future developments.

- Define generic support tools, and make such tools part of the new utility architecture.

C-NICE: OBJECTIVE AND APPROACH

The objective of the architecture is to facilitate the development of applications in computational mechanics. Within this broad field we emphasize the simulation of finite-element-based dynamics and control of space structures on concurrent multiprocessors.

NICE-type architectures for finite element machines on sequential machines are well developed and gaining acceptance. The support of finite element computations on the new concurrent multiprocessors raises two questions: (1) Can the sequential machine architecture be suitably extended to encompass concurrent machines, and (2) Do we need a totatly separate architecture for each machine type?

The approach taken is to try to extend selective components of the seqntial NICE architecture. A rewritten segment should (a) try to aniticipate "winners" in the concurrent machince race; (b) emcompass sequential computers naturally, and (c) be a significant improvement over the previous version even on sequential computers. Two components that would benefit substantially from a rewrite are the source code distribution utility MAX, and the input-output manager DMGASP. Therefore they were selected as the first two candidates for rewrite.

The expanded and revised MAX and DMGASP are called TIM and CPIO, respectively. TIM is a true source code preprocessor with knowledge about concurrent machine hardware and operating systems. CPIO is a concurrent paged I/O manager that can handle parallel I/O. Progress in these two components is summarized on the last slide.

THIS PAGE LEFT BLANK INTENTIONALLY

C-NICE: PROGRESS

- *Source Code Preprocessor.* A generalized source code preprocessor, TIM, has been coded as a replacement to the NICE tool MAX. The input to TIM is a blocked file. TIM recognizes code blocks in Fortran 77, Force, CWEB and Assembly, TeX documentation blocks, data blocks and database records. It knows about 56 computer (8 concurrent ones) and several operating systems. It generates output file by interpreting very high level expressions (for example: the target system is Alliant) and macro substitution. Handles arbitrary logical expressions and macro substitution.

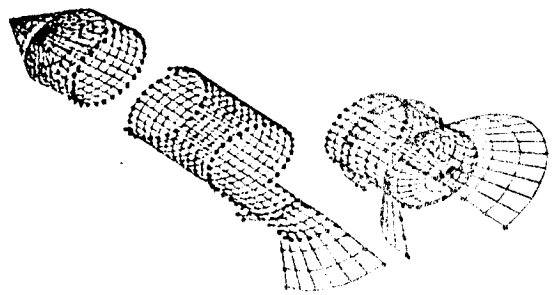
- *Concurrent Paged I/O Manager.* CPIO has been designed as a successor to the I/O Manager DMGASP of the NICE system. Accepts C and C++ data structures, and allows simultaneous access to logical devices. Logical devices may use parallel disk I/O. Implementation will be based on paged buffer pool serving concurrent processes. It should be also more efficient than the old IOM on sequential machines.

C-NICE: FUTURE PLANS

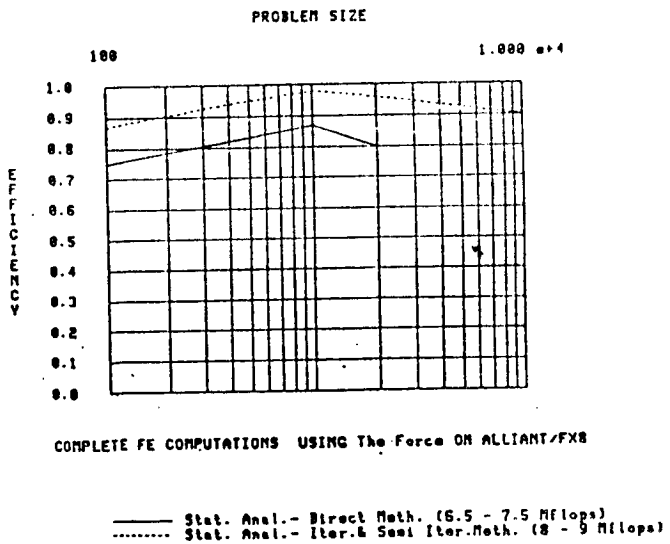
- *Finite Element Applications.* We shall continue experimentation on the Cray 2 multitasking computer at NASA Ames, helped by a Cray version of Harry Jordan's The Force preprocessor. We plan eventually to experiment with finite element calculations on the ETA10 (under CDC sponsorship) and the Connection machine (under NRL sponsorship).

- *Architecture Tools.* Document and release the TIM programming tool. Implement CPIO on Sun, Alliant, Cray 2/XMP MT, and the Connection Machine.

CONCURRENT FE COMPUTATIONS - PROGRESS



- Development of a Practical Automatic FE Domain Decomposer.
- Development of a Coloring Scheme for Explicit Element-by-Element Computations.
- Procedures for Dynamic Re-Mapping of Processors.
- Design, implementation and testing of a complete prototype Concurrent Static FE Analyzer using FORCE.
- Test versions available for:
  Cray2 with UNICOS Multitasking
  Cray X-MP with UNICOS Multitasking
  Alliant FX/8
  Encore Multimax
  Sequent-Balance



CONCURRENT COMPUTATIONS VIA 'FORCE' - PROGRESS

A software architecture that parallelizes finite element computations in their totality has been designed and tailored to several different multicomputing environments. Based on the *divide* and *conquer* paradigm, the selected approach has shown the potential of handling various parallel numerical schemes for different purposes.

An automatic domain decomposer was designed and implemented on several multiprocessors. It meets three basic requirements: (1) it handles irregular geometry and arbitrary discretization pattern in order to be general-purpose; (2) it delivers a set of balanced subdomains in order to ensure that the overall computational load will be as evenly distributed as possible among the processors; (3) it minimizes the amount of interface mesh nodes because these can become burdensome in communication and/or synchronization requirements.

On a shared memory multiprocessor, internal boundary elements constitute a source of potential memory conflict during parallel computations that are carried out on the element level. To eliminate the need for serializing computations in these critical regions, a graph coloring type of algorithm was devised and applied to the interface of the subdomains. A general framework for parallel Element-by-Element explicit computations was also devised. Based on the coloring scheme mentioned above, it minimizes the amount of *Critical Sections* and requires only one fork/join procedure per global iterative and/or time step.

In order to achieve load balance in fully nonlinear computations, two different processor mappings are combined and dynamically activated: (1) for the element level explicit computations and (2) for the global level implicit computations.

A prototype FE static code for shared memory super multiprocessors is now available. Several numerical algorithms were parallelized and revised to connect the software architecture with the finite element solution algorithms. These include an Active Column equation solver, a Block SOR Iteration, a Block Asymmetric Factorization, and a Preconditioned Conjugate Gradient algorithms. Vectorization is achieved within each concurrent process.

Portability of the code is guaranteed through "The Force"

Center for Structures and Controls, University of Colorado, Boulder, CO

RESEARCH APPROACH: Multibody Dynamics

- Formulate Equations of Motion for System Components – Flexible Elements, Joints, Rigid Elements, Constraints, Control Systems.
- Develop Computational Algorithms for Large-Scale Computations of Each of the System Components.
- Develop Concurrent Partitioned Analysis Procedures for Overall Multibody Dynamics Analysis.
- Implement the Formulations, Solution Algorithms and Partitioned Analysis Procedures and Perform Large-Scale Simulation of System-Level Dynamics Problems.

RESEARCH APPROACH: Multibody Dynamics

In the existing multibody formulations, the minimal achievable number of equations has been the driving factor in the derivation of the equtions of motion. For example, the *Order-N* formulation is a special form suited for open-loop connectivity. Such formulations necessarily require to integrate the constraints, joints, rigid elements *in their entirety* into the equations of motion. Such approach, while compact in the equations of motion and efficient for open-end manipulator dynamics, become unwieldy for deployment of lattice truss structures.

In the present approach, we formulate each of the multibody dynamics elements – flexible beams, joints, constraints, rigid bodies, and control forces– as a seperate entity. We separately solve the equations motion for beams, rigid elements, constraint forces and control forces. The system-level dynamic response is obtained by applying the partitioned analysis procedures. When found advantageous, we can reduce part of the system equations through system topology to a set of *Order-N* equations.

We believe that our approach is well suited not only for sequential computations while preserving software modularity but particularly appeals to parallel computations as each of the component solutions can be carried out in parallel.

Equations of Motion for MBD

$$\delta F^I + \delta F^S + \delta \underline{\Phi}^J \cdot \lambda_J + \delta \underline{\Phi}^C \cdot \lambda_C = \delta F^E$$

Inertia: $\delta F^I = \delta \underline{u}^T \cdot \underline{\underline{M}} \cdot \ddot{\underline{u}} + \delta \underline{\alpha}^T \cdot \underline{\underline{J}} \times \dot{\underline{\omega}} + \delta \underline{\alpha}^T \cdot \underline{\underline{J}} \times \underline{\omega} \cdot \underline{\omega}$

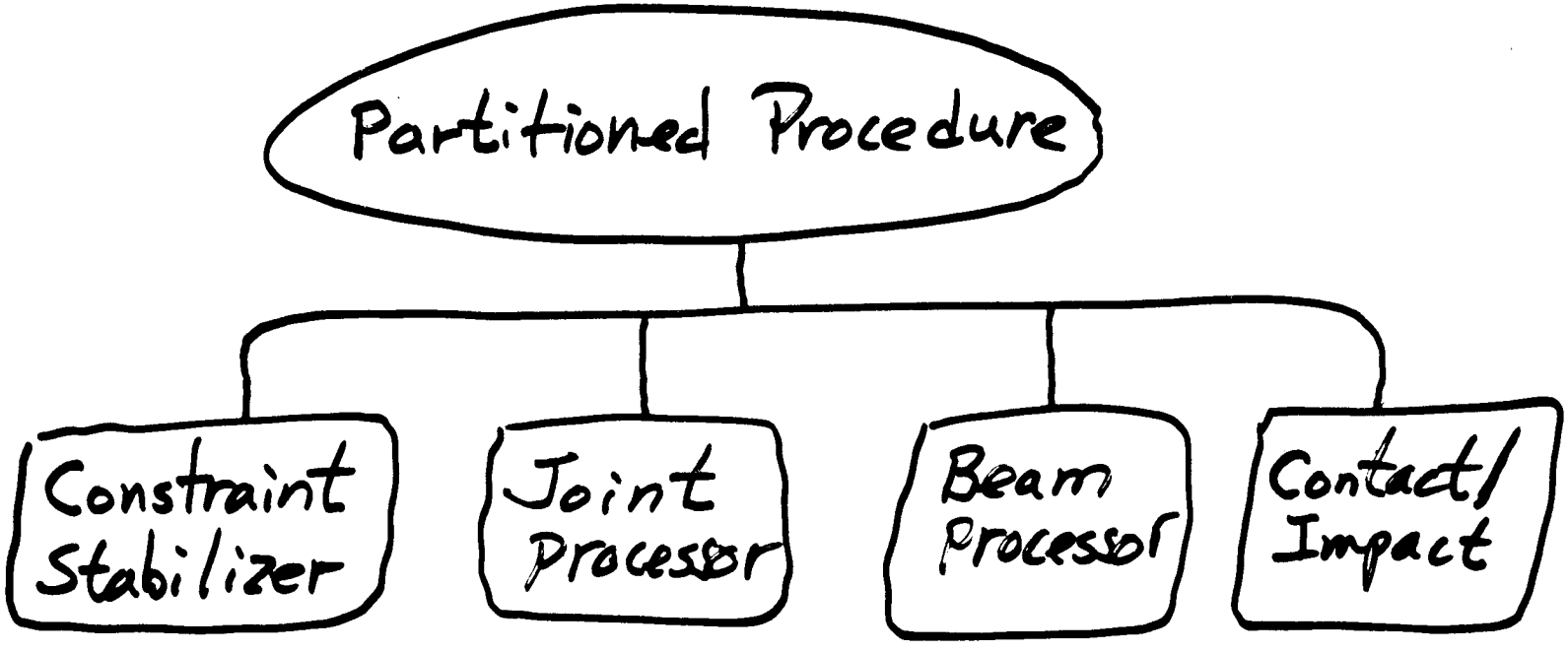
stiffness: $\delta F^S = \delta u \cdot \underline{n}_I + \delta \alpha^T \cdot \underline{M}_I$

Joints: $\delta \underline{\Phi}^J = 0$

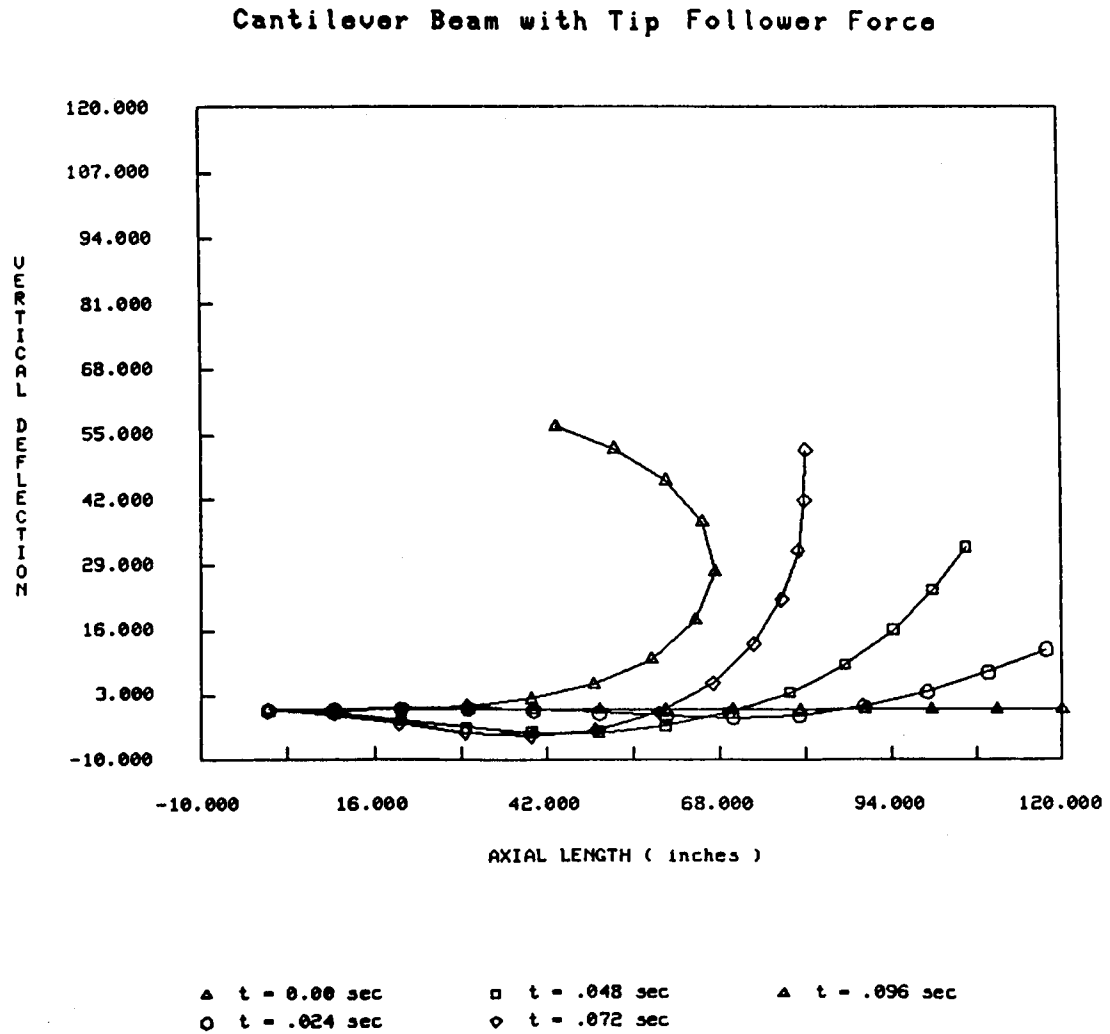
Constraints: $\delta \underline{\Phi}^C = 0$

External: $\delta F^E = 0$

Implementation of MBD



RESEARCH PROGRESS: Multibody Dynamics
(Flexible Beams Undergoing Large Motions)



RESEARCH PROGRESS: Multibody Dynamics
(Flexible Beams Undergoing Large Motions)

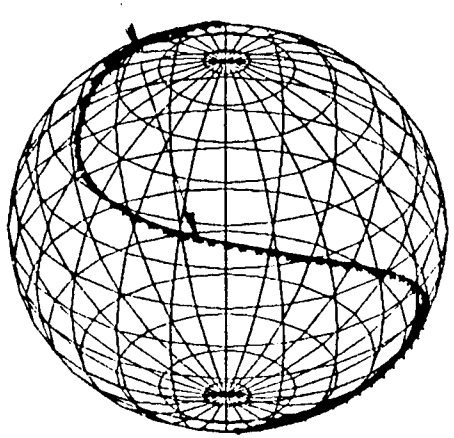
A beam formulation that admits large rotational and translational motions has been developed. The beam formulation can be implemented with and/or without transverse shear strains.

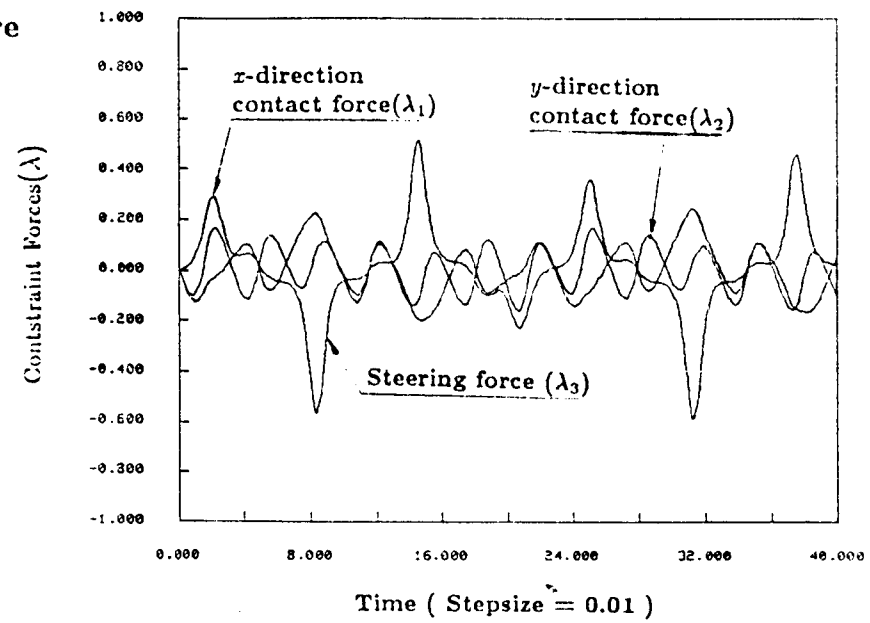
Its Specific Features Are:

- When flexibility is negligible, classical rigid-body equations are recovered.
- Translational and angular motions are measured from the inertial and corotational frames, respectively; that is, no coupling between the translational inertia and rotational inertia results.
- Strains are computed directly from total displacements and finite rotation variables; no limitation on the relative magnitudes between rigid and flexible motions is required.

RESEARCH PROGRESS: Multibody Dynamics
(Constraint Stabilization and Large Rotation Algorithm)

Sinusoidal Ball Track Mapped on the Sphere





Sphere with Off-Set Center Rolling on a Sinusoidal Curve

RESEARCH PROGRESS: Multibody Dynamics
(Constraint Stabilization and Large Rotation Algorithm)

A stabilization technique for accurately incorporating both the configuration and motion constraints into the multibody dynamics formalism has been developed. In addition, a computational procedure for updating large rotational motions has been developed.

Specific Features Are:

- It overcomes singularity difficulty when constraints become linearly dependent within computational precision.
- It facilitates a modular solution package for the constraints independent of the solver of the equtions of motion.
- It consistently yields more accurate solutions than the Baumgarte technique for problems tested so far.
- The large rotational update algorithm computes the translational motions by an explicit integration technique whereas the rotational motions are treated by an implicit integration via the Euler parameters.

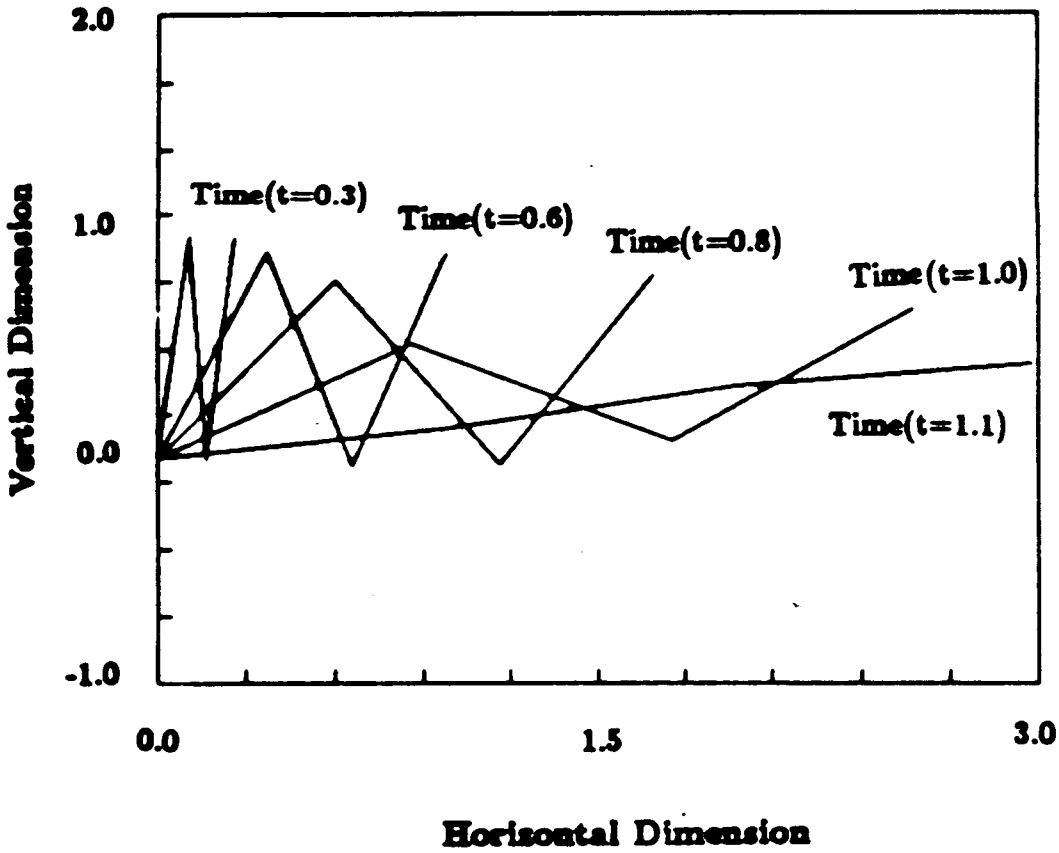
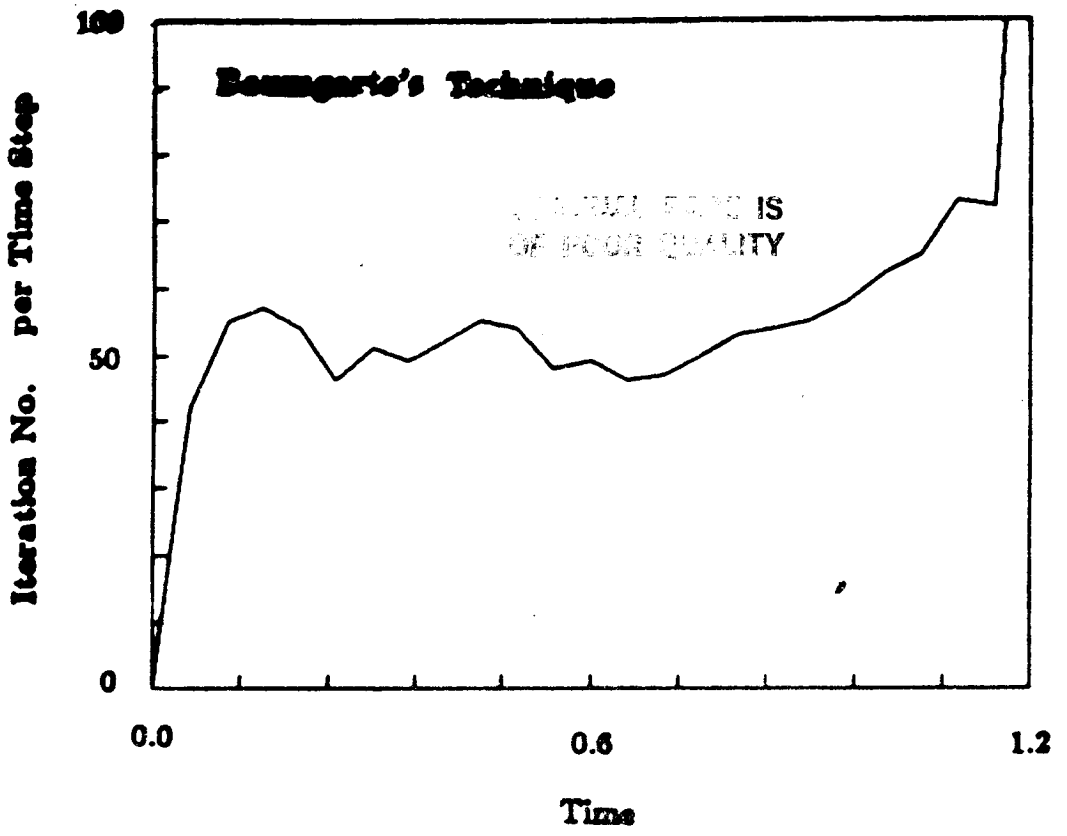


Fig. 5 Deployment of Three-Link Remote Manipulator



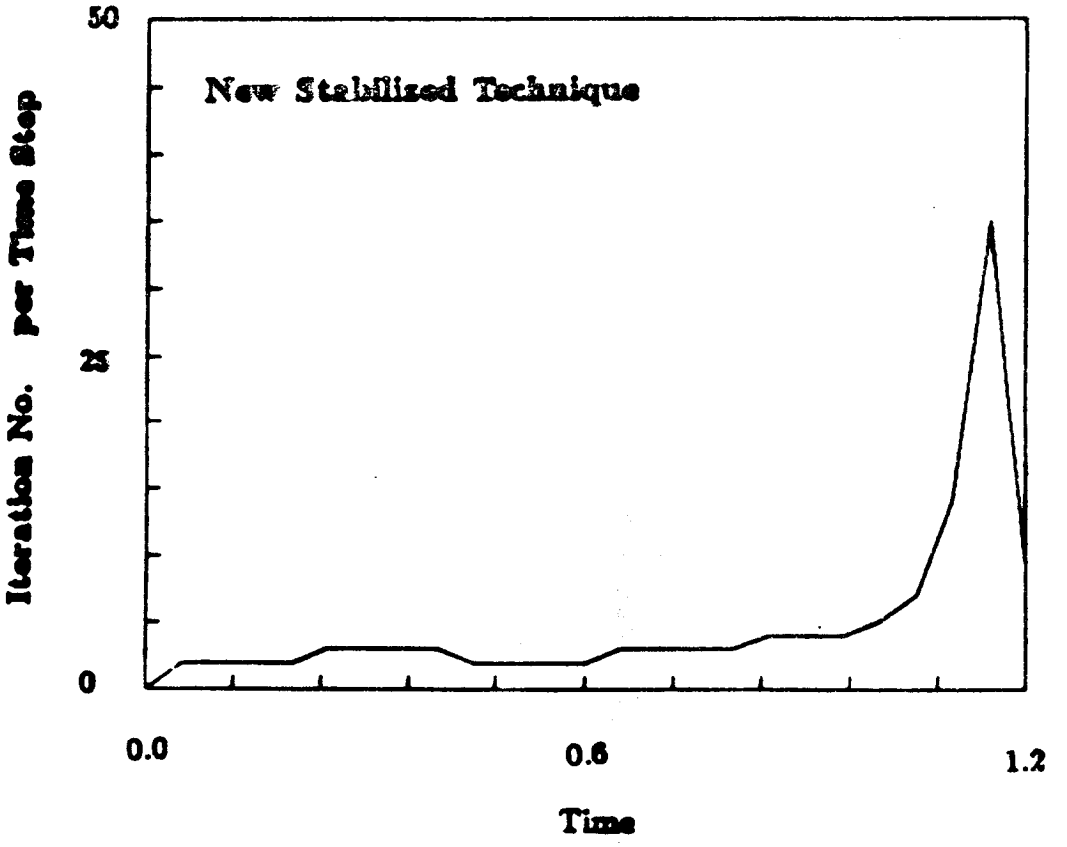


Fig.6 Performance of Two Stabilization Techniques for Three-Link Remote Manipulator (Solution Accuracy=10^{-6})

Related Research Activities to CSM at CU

- Shell Dynamics (NRL) - Rely on CSM shell processor
- Control-Structure Interaction (AFOSR) - Partitioned analysis leads to modular interaction analysis (NASA (W.K. Belvin, J. Juang) - CU Collaboration.)
- ETA 10 Concurrent Computations (CDC) - Parallel Software developed is used to test C-NICE I/O concepts
- Connection Machine Experiment (DARPA/NRL) - A Parallel computational Mechanics Experiment

Future Plans (12/87 – 11/88)

- C-NICE :
  - Implement A First Version of Parallel I/O
    - Parallel Control for CRAY-II & Conn. M/C.
    - Source Code Management System (TIM)
- Concurrent Applications Software Tools :
  - Further Experiments with FORCE/CRAY-II UNICOS Multitasking to solve nonlinear dynamics problems.
- Multibody Dynamics :
  - Transform research modules of constraint solver, flexible beams, joints, topology mapping schemes, partitioned procedure into an experimental processor & attempt to implement it into CSM/Testbed.

Future Plans – Cont'd

- Multibody Dynamics :
  - Implement the 5 modules on the Alliant/8 and refine partitioned procedures.